VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

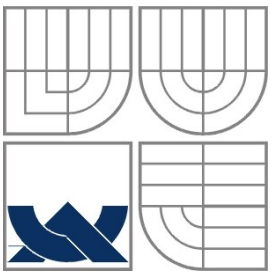
MONITORING HYBRIDNÍ SÍŤOVÉ INFRASTRUKTURY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

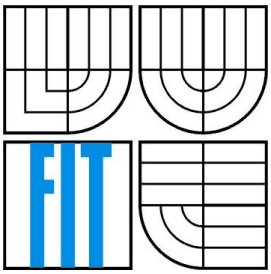
AUTOR PRÁCE
AUTHOR

MARTIN JANDA

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MONITORING HYBRIDNÍ SÍŤOVÉ INFRASTRUKTURY

MONITORING HYBRID NET INFRASTRUCTURE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN JANDA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2009

Abstrakt

Tato bakalářská práce se věnuje vývoji univerzálního monitorovacího systému zařízení v hybridní síti pro společnost SELF servis. Systém slouží k ověření dostupnosti pomocí ICMP-paketů a ke sledování hodnot s využitím SNMP protokolu.
V textu naleznete popis analýzy požadavků, návrhu systému a samotné implementace. V závěrečné kapitole se věnuji výsledkům z reálného provozu a nápadům pro další vývoj.

Abstract

This bachelor thesis follows developement of automatic monitoring system of devices in hybrid network for SELF servis company. System's purpose is to check availability by ICMP packets and to watch values by SNMP protocol. In this text you'll find description of requirements analysis, design of the system and implementation. In the last chapter follows results from real use and ideas for further development.

Klíčová slova

monitorování, monitoring počítačové sítě, monitorovací systém, snmp, icmp, sql, postgresql, linux, php, perl.

Keywords

monitoring, computer network monitoring, monitoring system, snmp, icmp, sql, postgresql, linux, php, perl.

Citace

Martin Janda: Monitoring hybridní síťové infrastruktury, bakalářská práce, Brno, FIT VUT v Brně, 2009

Monitoring hybridní síťové infrastruktury

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. RNDr. Pavla Smrže, Ph.D. Další informace mi poskytl Mgr. Radek Bednář ze společnosti SELF servis. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Janda
20. května 2009

Poděkování

Děkuji vedoucímu své bakalářské práce, panu Doc. RNDr. Pavlu Smržovi, Ph.D. za cenné připomínky. Dále bych chtěl poděkovat Mgr. Radku Bednářovi a společnosti SELF servis za umožnění realizace tohoto projektu.

Obsah

Obsah....1
1 Úvod....3
2 Analýza požadavků....4
2.1 Současný stav....4
2.2 Požadavky na funkčnost systému....5
2.3 Dostupná Open source řešení....5
2.3.1 Nagios....6
2.3.2 Centreon....6
2.3.3 Cacti....6
2.4 Vývoj vlastního systému....6
2.5 Dekompozice zadání....7
2.6 Volba technologie sledování zařízení a získávání dat....7
2.6.1 Protokol ICMP....7
2.6.2 Protokol SNMP....7
2.7 Volba programovacího jazyka....9
2.7.1 Webové rozhraní....9
2.7.2 Kolektor - sběrač dat....10
2.8 Volba datového úložiště....11
2.9 Shrnutí analýzy....12
3 Návrh....13
3.1 Interakce prvků systému....13
3.2 Kolektory....14
3.2.1 Skript kolektor_ping....14
3.2.2 Skript kolektor_snmp....14
3.3 Návrh databáze....15
3.3.1 Tabulky....15
3.3.2 Triggery....18
3.3.3 Uložené procedury....19
3.3.4 Řešení problému velkého objemu dat v tabulce statistics....20
3.4 Webové rozhraní....21
3.4.1 Pohledy a funkce pro uživatele....21
3.4.2 Pohledy a funkce pro administrátora....22
3.5 Použití knihoven při vývoji webového uživatelského rozhraní....23
3.5.1 PHP framework Nette....23
3.5.2 Dibi – knihovna pro práci s databází....24

3.5.3 JPGraph – knihovna na kreslení grafů....24
3.5.4 JQuery – JavaScriptová knihovna....24
4 Implementace....26
4.1 Skripty kolektoru....26
4.1.1 Implementace skriptu kolektor_ping.pl....26
4.1.2 Implementace skriptu kolektor_snmp.pl....26
4.2 Uživatelské webové rozhraní....27
4.2.1 Implementace autentizace a autorizace....28
5 Nasazení a provoz systému....29
5.1 Instalace....29
5.1.1 Požadavky na provoz databáze....29
5.1.2 Požadavky a provoz kolektorů....29
5.1.3 Požadavky a provoz webového rozhraní....29
6 Závěr....31
6.1 Zhodnocení provozu....31
6.2 Další vývoj systému....31
6.2.1 Krátkodobé plány....31
6.2.2 Dlouhodobé plány....32
6.3 Osobní přínos....32
Literatura....34
Seznam příloh....36

1 Úvod

Počítačové sítě už 40 let ovlivňují a mění náš každodenní život i práci. Z původně vojenského projektu decentralizované sítě ARPANET realizované v roce 1969 v USA k dnešní podobě Internetu vedl překotný technologický vývoj, který pokračuje i dnes. Počet uživatelů Internetu roste téměř exponenciálně a v roce 2008 bylo k Internetu připojeno odhadem 1,5 miliardy uživatelů [1].

S počtem uživatelů roste i počet poskytovatelů připojení (dále jen ISP – Internet service provider). S postupnou digitalizací televizního vysílání a rozšíření využití hlasových služeb v počítačových sítích dnes ISP často kromě samotného připojení k Internetu nabízí i zmíněné služby v rámci jedné infrastruktury. To vede k potřebě vybudovat a udržovat robustní počítačovou síť. Jedním z požadavků chodu robustní počítačové sítě je **monitorování provozu**.

Při monitorování počítačové sítě z pohledu ISP je důležité si ujasnit proč a co sledovat. Jako hlavní důvody lze uvést:

- sledování provozu a vytížení sítě (aktuální stav a historie objemu provozu),
- sledování zařízení v síti (včasná detekce poruchy),
- zúčtovatelnost nabízených služeb klientům (záznam o objemu konektivity, dostupnost služby atd.).

Z jiného úhlu pohledu mohu říci, že provozovateli jde jak o aktuální informace a stav sítě, tak o statistické vyhodnocení a historii provozu a událostí ve sledované síti.

Cílem mé bakalářské práce je najít, navrhnout a implementovat vhodné univerzální řešení monitorovacího systému s jednotným úložištěm dat pro potřeby společnosti SELF servis, spol. s r.o. Zadání a především řešení je v dalších kapitolách podrobněji rozvedeno. Samotná textová část práce se pak v jednotlivých kapitolách věnuje analýze požadavků na monitorovací systém, hledání dostupných řešení, návrhu vlastního systému, popisu implementace a vyhodnocení reálného provozu. Práce končí návrhy pro další vývoj a zhodnocením jednotlivých fází projektu.

Text je určen správcům sítě, kteří hledají řešení monitoringu jejich sítě. Kromě případu použití samotného vzniklého systému, může být tato práce užitečná i jako podklad k analýze a jako upozornění na problémy, které se objevily při vývoji. Další skupinou jsou studenti hledající příklad komplexnějšího využití databázového systému.

2 Analýza požadavků

Prvotním impulzem byla potřeba vyřešit monitorování a statistiky počítačové sítě ve společnosti SELF servis. V tu chvíli ještě nebylo rozhodnuto o implementaci vlastního řešení, ale tato analýza specifikovala základní formulaci požadavků na výsledný systém. Pro úplnost uvedu porovnání dostupných řešení a vlastní implementace a zmíním důvody vedoucí k rozhodnutí vlastní realizace monitorovacího systému. Při analýze jsem pro získání informací použil rozhovory, analýzu existujícího řešení a osobní zkušenosti z přímé účasti v pracovním procesu.

V úvodní části analýzy čtenáře seznámím s prostředím společnosti. Společnost SELF servis je jedním z předních ISP v ČR. Zaměřuje se na poskytování datové konektivity a nabízí Internet, digitální televizi a telefonní služby pro domácnost. SELF servis provozuje hybridní páteřní síť pro vysokorychlostní datové přenosy pod názvem CzechBone.

Z pohledu fyzické a linkové vrstvy referenčního modelu ISO/OSI se síť skládá z LAN sítí typu Ethernet spojených optickými vlákny v jednu WAN síť tvořenou VLAN sítěmi. Na vyšších vrstvách se využívá klasická skladba Internetových protokolů.

2.1 Současný stav

Kromě samotných požadavků na funkčnost monitorovacího systému zmíním současné způsoby monitorování sítě. Z nedostatků stávajícího řešení vyplývá řada požadavků na nový monitorovací systém.

V tuto chvíli se používá několik nezávislých a oddělených nástrojů. Pro aktuální zjištění stavu zařízení v síti se používá nástroj NetSaint [2], který umožňuje kontrolu dostupnosti zařízení a funkčnosti služeb běžících na serveru. Pro sběr statistik a vytváření grafů se používá několik různých vlastních implementací s použitím knihoven a nástrojů RRD-Tool [3] a NET-SNMP [4]. Tyto skripty využívají protokol SNMP pro periodický sběr informací ze síťových zařízení (switche, routery, atd.), které následně ukládají do databáze typu RRD. U některých implementací běží druhý skript, který periodicky vykresluje grafy nebo se o vykreslení stará CGI skript v okamžiku požadavku uživatele přes HTTP. Tento systém je, s drobnými úpravami, používán již řadu let.

Podstatné nedostatky současného řešení jsou:

- roztříštěnost a různorodost,
- obtížnější konfigurace, zdlouhavé zavádění nových zařízení do monitorování,
- malé možnosti dynamické tvorby grafů (pevně nastavené v konfiguraci),
- zastaralost použitých knihoven,

– přetěžování daných serverů s nárůstem a rozrůstáním sítě, komplikovaný přenos zátěže na jiný server.

Z pohledu lidských zdrojů je nevyhovující časová náročnost správy těchto systémů. Při úpravě je totiž potřeba mít přístup na daný server, mít k dispozici dokumentaci s popisem jak zařízení zavést a ručně vytvořit patřičné konfigurační soubory. Dalším zjištěným nedostatkem je nemožnost tvorby dynamických grafů.

2.2 Požadavky na funkčnost systému

Z analýzy stávajícího stavu monitoringu ve společnosti SELF servis a z rozhovorů s Mgr. Radkem Bednářem a dalšími spolupracovníky vyplynuly následující požadavky na nový systém:

1. test dostupnosti zařízení a sběr dat přes SNMP
2. možnost postupného rozšiřování
3. sledování aktuálního stavu zařízení a tvorba statistických grafových a tabulkových přehledů
4. přístup přes webové rozhraní k statistikám i konfiguraci
5. pohodlné a rychlé přidávání zařízení a jejich konfigurace
6. rozšiřitelnost systému – jednotné úložiště dat při zachování možnosti využití více serverů pro sběr dat
7. flexibilní tvorba grafů dle potřeby uživatele – například denní pohledy i v historii, zobrazení součtů a rozdílů hodnot z různých zařízení, apod.
8. možnost řízení přístupu a práv uživatelů přistupujících k systému.

2.3 Dostupná Open source řešení

Jak už bylo řečeno v úvodu kapitoly – potřeba analýzy požadavků vznikla ještě před rozhodnutím vytvořit vlastní systém. Požadavek č. 2 určuje potřebu zvolit buď vlastní vývoj systému nebo využít jeden z dostupných Open source systémů pro monitoring a statistiky. V této podkapitole shrnu výsledky mého průzkumu dostupných řešení pod svobodnou licencí a uvedu stručné odůvodnění, proč jsem se rozhodl je nepoužít.

Výhodou všech dále zmíněných Open source řešení je pořizovací cena, otevřenost zdrojového kódu a pokračující vývoj komunitou.

Především je nutné říct, že je hledán jeden systém, který splní všechny dané potřeby. Analýza nástrojů proběhla v říjnu roku 2008. Sumarizací požadavků jsem k vyzkoušení vybral systémy Nagios, Centreon a Cacti.

2.3.1 Nagios

Nástupce systému NetSaint poskytuje opravdu širokou nabídku možností sledování zařízení a serverů v síti včetně kontroly dostupnosti běžících služeb. Bohužel nesplňuje v plné šíři požadavek na tvorbu dynamických grafů. Z osobního pohledu mi dále nevyhovovala náročná konfigurace a nepřehlednost webového rozhraní.

2.3.2 Centreon

Monitorovací systém Centreon vychází ze zmíněného systému Nagios. Z požadovaných funkcí jsem nepřišel na způsob tvorby dynamických grafů součtů z více zařízení. Nedostatkem je také nekompletní dokumentace v angličtině.

2.3.3 Cacti

Dalším z testovaných Open source řešení je Cacti. Jako nevýhodu jsem zjistil, že nelze pohodlně rozdělit zátěž sběru dat na více serverů a dále rovněž nenabízí tvorbu dynamických grafů. Z osobní zkušenosti s Cacti musím zmínit, že se mi nepovedlo korektně zprovoznit skript pro sběr dat ani po pročtení dokumentace.
Testované Open source nástroje nesplnily všechny přednesené požadavky. Navíc jsem zjistil několik nedostatků z osobní zkušenosti při testování.

2.4 Vývoj vlastního systému

Vývoj vlastního monitorovacího systému přináší 2 podstatné výhody:

- řešení přesně na míru a splnění všech požadavků,
- možnost přepracování a dalšího vývoje dle aktuálních potřeb.

Je třeba však zmínit, že zde existuje také zásadní nevýhoda a to, že vývoj stojí čas a lidské zdroje. Při plánování je potřeba zohlednit délku vývoje, možnosti a zkušenosti vývojáře, případně vývojářů.

Výhody vývoje vlastního systému při rozhodování převážily nad nevýhodami. Bylo rozhodnuto vytvořit vlastní systém pro monitoring zařízení v síti a tento projekt je náplní mé bakalářské práce. Pracovní název systému je NetSurveyor.

2.5 Dekompozice zadání

Požadavek rozšiřitelnosti systému mě přivádí k nutnosti rozdělit celý systém do více části, které mohou pracovat odděleně. Vznikají zde tak 3 podskupiny funkčnosti systému:

1. Kolektory – aplikační část, která se stará o získávání informací o zařízení a ze zařízení. Kolektorů může běžet 1 až n, čímž lze rozdělit zátěž mezi více serverů.
2. Úložiště dat – server, kam jsou ukládány sesbíraná data a skladovány pro pozdější statistické vyhodnocení.
3. Webové uživatelské rozhraní – část aplikace, která může běžet také na jiném serveru. Její hlavní úlohou je interakce s uživatelem. Konkrétně pak zpřístupnění aktuálního stavu, statistických výsledků a možnost konfigurace.

2.6 Volba technologie sledování zařízení a získávání dat

V této kapitole se zaměřím na analýzu a volbu technologie pro sledování zařízení v sítích a získávání informací z nich. Vybranou technologii stručně popíši.

2.6.1 Protokol ICMP

Jak uvádí Dostálek [6] „protokol ICMP je služební protokol, který je součástí IP-protokolu. ICMP slouží k signalizaci mimořádných událostí v sítích postavených na IP-protokolu. ICMP-paket „Žádost o echo“ je žádost o odpověď poslaná k zařízení.“ Jedná se o nejčastější způsob ověření dostupnosti zařízení v síti. V různých OS bývá tento způsob dostupný pomocí aplikace ping. ICMP je specifikované v RFC 792 [7].

Na žádost o echo odpovídají i „hloupější“ prvky a zařízení v síti. Jedná se o zdrojově nenáročný způsob otestování dostupnosti, který použiji i já v mé práci.

2.6.2 Protokol SNMP

V RFC 1157 [8] je protokol SNMP (Simple Network Management Protokol) definován jako protokol a architektura pro správu sítě. Účel protokolu je komunikace mezi zařízením (označován jako agent) a řídícím systémem (označován jako network management station) za účelem získání stavových informací a řízení agenta. Protokol SNMP byl vytvořen s důrazem na jednoduchost implementace na straně sledovaného zařízení.

Protokol SNMP pracuje nad transportním protokolem UDP a nejčastěji používá porty 161 a 162. Samotný SNMP protokol existuje ve 3 verzích, přitom tou nejrozšířenější verzí je verze 2.

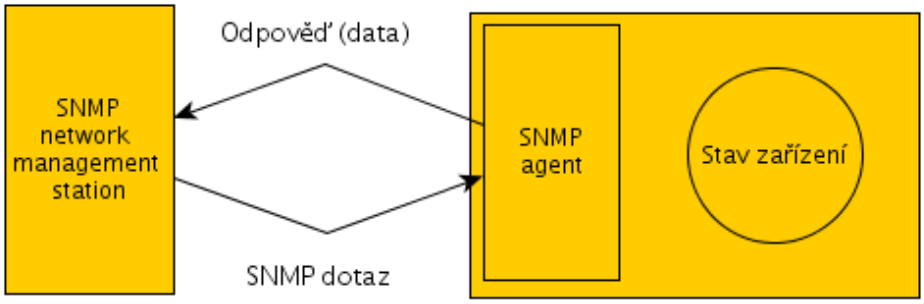
První verze byla specifikována v RFC 1157 [8] a nabízela pouze základní autentizaci. V dnešní době se používá minimálně.

SNMPv2, tedy druhá verze definovaná v RFC 1441 [9] přinesla několik vylepšení. Především se jednalo o možnost hierarchického monitorování a usnadnění práce s daty (byl například přidán typ pro 64 bitový čítač). Verze 2 je i v současné době nejrozšířenější. Z důvodu slabé bezpečnosti bývá SNMPv2 častěji používán pouze pro čtení dat, nikoliv ke konfiguraci síťového zařízení jak mohu z osobní zkušenosti potvrdit.

Protokol SNMPv3 specifikován v roce 2002 v RFC 3411 [10] rozšiřuje především možnosti kompletního šifrování včetně posílaných dat a také možnost bezpečnější autentizace. Bohužel je náročnější na implementaci na straně agenta a velká část používaných zařízení jej nepodporují.

Z důvodu standardizace a velkého počtu informací, které lze z agentů vyčíst jsou tato data organizována do adresovatelné struktury MIB (Management Information Base). MIB má hierarchickou stromovou strukturu, ve které je možné požadovaná data adresovat pomocí OID (Object ID). V MIB stromu jsou některé podstromy pevně definované podle RFC 1213 [11] a poskytují informace o daném systému, síťových rozhraní, apod. Dále existují rezervované podstromy pro některé větší společnosti a jejich zařízení. Vybrané firemní definice MIB jsou veřejně dostupné, jiné jsou zpoplatněny a chráněny licencemi. V MIB existují podstromy pro libovolnou vlastní implementaci.

MIB jsou nadefinovány a popsány pomocí podmnožiny ASN.1 (Abstract Syntax Notation), neboli notace pro abstraktní zápis struktur dat. ASN.1 je definováno v RFC 3641 [12].



Obrázek 1: Proces komunikace s využitím SNMP

Samotný proces komunikace probíhá tak, že manager pošle dotaz agentovi. Dotaz obsahuje autentizační údaje, příkaz operace a jednu nebo více OID informací, která managera zajímá. Pokud agent tuto informaci má a autentizace proběhne v pořádku, pošle požadovaná data.

Příkazů operací existuje více. Zde stručně popíši operace používané a specifikované v SNMPv2 [9]:

- get – dotaz vyžádání dat od agenta,
- getnext – požadavek na získání následující informace, tedy sousedního OID,
- walk – příkaz zřetězující volání get pro adresovaný podstrom,
- set – příkaz zapíše konkrétní data na agenta.

Další možností komunikace s využitím SNMP je zasílání a příjem tzv. TRAP signálů. Agent při určité situaci zasílá TRAP na předdefinovanou adresu managera. TRAP signály se používají především v situaci, kdy dojde k události, která si žádá pozornosti správce sítě.

Stejně jako protokol ICMP pro ověřování dostupnosti zařízení v síti nemá protokol SNMP pro tuto svou funkcionalitu rozumnou alternativu a vzhledem k rozšíření a podpoře ze strany síťových zařízení je SNMP zřejmá volba pro využití v mém projektu.

2.7 Volba programovacího jazyka

Při výběru programovacího jazyka jsem zohlednil tyto aspekty:

1. základní znalost jazyka z mé strany,
2. dostupnost knihoven a nástrojů usnadňující a urychlující vývoj,
3. platforma dostupná pro OS GNU/Linux..

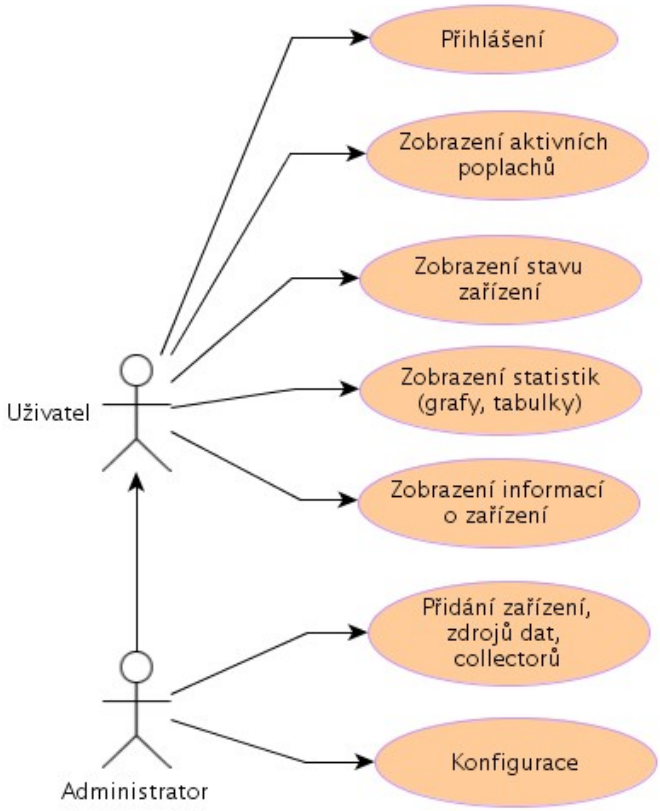
Dekompozice celého problému na 3 samostatné části umožňuje každou komponentu vyvíjet v jiném programovacím jazyku, který může být vhodnější, případně nabízí knihovny, které vývoj zjednoduší. V prvotní fázi projektu jsem se však snažil najít jeden programovací jazyk, ve kterém bych vyvíjel webové rozhraní i kolektor. Tímto jazykem se stal Perl.

2.7.1 Webové rozhraní

Jak již bylo řečeno, webové rozhraní zastává 2 stěžejní úkoly:

1. poskytnout informace o aktuálním stavu a přístupu k statistickému zobrazení naměřených hodnot,
2. umožnit pohodlné vkládání nových zařízení a definici sledovaných hodnot nebo konfigurace.

K webovému rozhraní se váže potřeba definovat uživatele. V první verzi systému budou pouze 2 typy uživatelů. Uživatel má pouze právo prohlížet statistiky, grafy a informace o zařízení a sledovaných hodnotách. Administrátoři pak navíc mohou vkládat nová zařízení, apod. Rozdělení pravomocí schematicky popisuje následující diagram užití systému:



Obrázek 2: Diagram případu užití systému

V případě administračních funkcí jde principiálně tedy o sérii formulářů zpřístupňujících řadu nastavitelných parametrů. Z pohledu zobrazení informací se jedná o grafické a tabulkové znázornění informací. Pro větší uživatelské pohodlí lze obecně řadu úkolů zautomatizovat, vytvořit průvodce, sestavit různé pohledy apod. Podrobnějšímu rozboru webového rozhraní se věnuji v kapitole 4.

Zde bych zmínil, že se později během samotného vývoje ukázalo, že jazyk Perl pro tvorbu webového rozhraní není příliš vhodný a vývoj spíš komplikuje než usnadňuje. Rozhodl jsem se tedy pro vývoj webového rozhraní použít jazyk PHP, který nabízí hned několik předností. Především se jedná o velice populární jazyk pro vývoj webových aplikací a z této popularity pramení velký počet dostupných knihoven a nástrojů, které lze při vývoji použít.

2.7.2 Kolektor - sběrač dat

Funkcionalita kolektoru spočívá především v periodické kontrole dostupnosti zařízení a načtení údajů ze zařízení přes SNMP a uložení těchto dat do vhodného datového úložiště. Pro splnění tohoto úkolu musí kolektor znát několik údajů. Tyto informace závisí na konfiguraci a mohou být uživatelem

měněny. Proto je vhodné ji rovněž mít uloženou v centrálním datovém úložišti a dynamicky si ji před provedením dotazů načítat.

Vzhledem k velmi častému opakování této akce je vhodné samotnou logiku kolektoru navrhnout co nejjednodušeji.

Další požadavek, který vyplynul z analýzy problému, je interval zjišťování dat. Potřeba je u různých zařízení a informací různá. Minimální možný interval však byl stanoven na 30 s.

2.8 Volba datového úložiště

Nejprve zopakuji požadavky, které se týkají datového úložiště. Především by mělo jít o jednotné úložiště sesbíraných dat i konfigurací. Dále má být umožněna tvorba dynamických grafů, což například znamená tvorbu součtu hodnot protočených oktetů na síťovém rozhraní jednoho zařízení se síťovým rozhraním jiného zařízení apod. Obecně lze tedy říct, že půjde o datové úložiště databázového typu.

Analýzou existujících nástrojů jsem zjistil 2 nejčastěji používané technologie:

1. RRD (Round Robin Database),
2. RDB (relační databázové systémy).

Model RRD představuje kompaktní způsob cyklického uložení dat, který lze stručně popsat takto - během inicializace databáze se stanoví maximální velikost datového souboru a při vyčerpání volného místa se začnou přepisovat nejstarší vložené údaje. Samotná realizace těchto operací bývá nejčastěji implementována pomocí Open source sady knihoven a nástrojů RRDtool [3]. Výhodou RRD je především pevně daná velikost uložených dat. Principiálně však chybí například možnost flexibilnější práce s daty, změna formátu dat a konfigurace „za chodu“, apod.

Gilfillan [13] tvrdí, že „Relační databáze umožňují vytváření vzájemných vztahů mezi libovolnými tabulkami prostřednictvím společných polí. Jedná se o velmi flexibilní systém a většina moderních databází je relační“. Vlastními slovy bych relační databázové systémy popsal jako systém zpřístupňující data formou tabulek, mezi kterými lze provádět operace relační algebry. Tyto operace se nejčastěji formulují strukturovaným dotazem jazyka SQL (Structured Query Language).

Vzhledem k dříve zmíněným požadavkům jsem zvolil pro realizaci datového úložiště relační databázový systém.

Požadavky na výběr databázového systému jsem si formuloval takto:

- podpora SQL,
- robustnost a spolehlivost i pro vysokou zátěž,
- nabízeno pod některou z Open source licencí,
- možnost vnitřně realizovat část aplikační logiky nad daty (uložené procedury),

– existující knihovny v PHP (webové rozhraní) a Perlu (kolektory)

Využití RDS sebou nese i tu výhodu, že se tím řeší komunikace mezi kolektorem a úložištěm dat. Relační databázové systémy mají vlastní implementaci vzdálené komunikace přes počítačovou síť.

Jako Open source je nabízena celá řada databázových systémů. Já jsem v užším výběru vybíral mezi MySQL a PostgreSQL.

V první fázi projektu padl můj výběr na MySQL 5, především z důvodu mé předchozí zkušenosti.

Později jsem při práci s MySQL narazil na problém vytváření uložených procedur. MySQL nabízí pouze vlastní SQL jazyk a vyjadřovací schopnosti tohoto jazyka se mi jevily jako nedostatečné pro mé potřeby. PostgreSQL oproti tomu nabízí výběr z celé řady jazyků pro tvorbu uložených procedur, například PLPQSL, PLPerl, PLPython [14]. Navíc jsem u jiného projektu měl možnost poznat, že důkladnější konfigurací a návrhem tabulek lze u PostgreSQL podstatně zlepšit výkonnost.

Nakonec jsem tedy zvolil použití databázového systému PostgreSQL.

2.9 Shrnutí analýzy

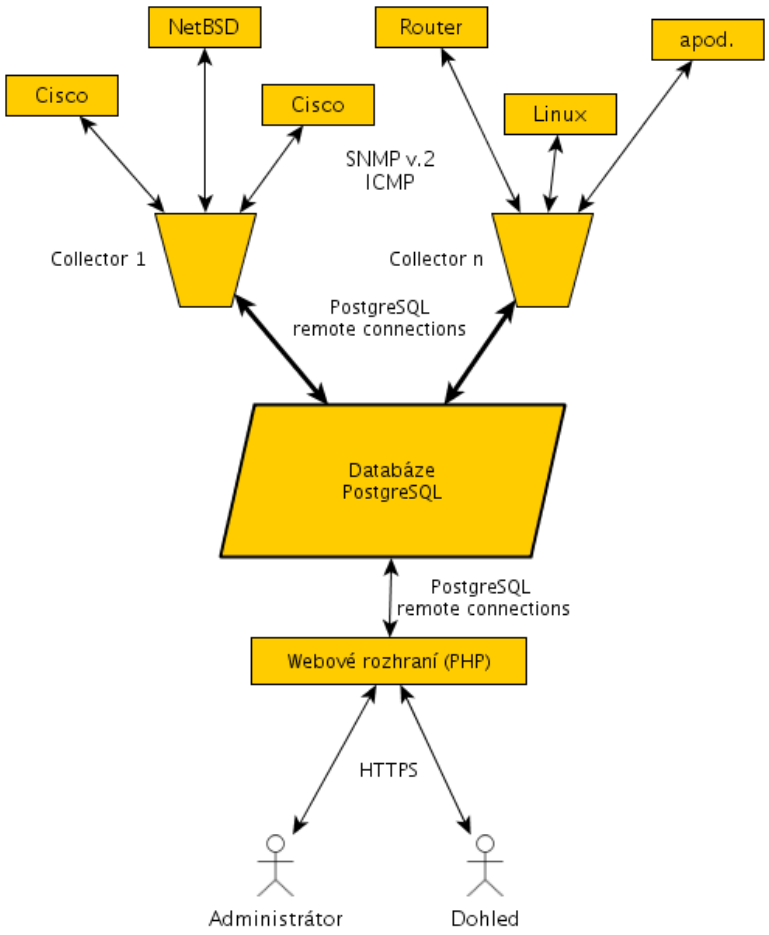
Analýzou požadavků na monitorovací systém jsem dospěl k závěru, že ke splnění všech zjištěných požadavků budu muset realizovat vlastního řešení. Dále jsem funkčnost systému vhodně rozdělil do 3 oddělených částí: kolektor, datové úložiště a webové rozhraní.

Jako programovací jazyk pro vývoj kolektoru, tedy aplikace která se stará o zjišťování stavu zařízení a sběr informací, jsem vybral Perl. Pro vytvoření webového rozhraní se v průběhu realizace ukázalo jako vhodnější použít jazyk PHP. Rovněž u řešení datového úložiště jsem v první fázi zvolil špatnou cestu a až později jsem našel vhodnější řešení ve formě použití PostgreSQL.

3 Návrh

V této kapitole se věnuji návrhu systému pro monitorování síťových zařízení. V úvodní části popíši vazby mezi komponentami systému, jejich vzájemnou komunikaci a směr toku dat. Dále předložím navrhovanou strukturu databáze a na závěr budu popisovat návrh aplikace kolektorů a webového rozhraní.

3.1 Interakce prvků systému



Obrázek 3: Komunikace mezi prvky systému

Obrázek 3 popisuje vzájemnou komunikaci mezi komponentami i komunikaci s okolím. Kolektory zasílají SNMP požadavky na různá zařízení, případně pouze pomocí ICMP paketu zkouší dostupnost daného zařízení. Přijaté informace vkládají pomocí vzdáleného připojení do databáze. Databáze kromě samotného úložiště bude obsahovat i část aplikační logiky a to takové, která se zabývá agregací a kontrolou dat. Například při vložení sledované hodnoty se zkontroluje, zda nedošlo

k překročení hranice pro vyvolání poplachu, apod. Webové uživatelské rozhraní se na základě požadavku dotazuje databáze a získává zpět patřičné informace, které vhodně zpracuje a předloží v podobě HTML stránky a případných grafů (v podobě bitmapových obrázků) zpět uživateli.

Z pohledu komunikace zde máme použití SNMP mezi zařízeními a kolektory, dále PostgreSQL remote connection spojující databází s kolektory nebo webovým rozhraním a komunikaci přes protokol HTTPS mezi aplikací webového rozhraní a prohlížeči uživatelů.

3.2 Kolektory

Jak už bylo řečeno, kolektor představuje aplikační část starající se o kontrolu dostupnosti zařízení a o sbírání dat pomocí SNMP protokolu. Tím, že je možné do systému přidat zařízení jen za účelem kontroly dostupnosti, mohu problém rozdělit a vytvořit 2 skripty kolektor_ping a kolektor_snmp. Aplikace kolektorů budou napsány v programovacím jazyce Perl.

3.2.1 Skript kolektor_ping

Smyslem skriptu je kontrola dostupnosti zařízení pomocí dříve popsaného ICMP-paketu. Algoritmus je zřejmý: v periodickém intervalu načítá skript údaje o zařízení z databáze a testuje zda odpoví. Výsledek je zapsán zpět do databáze. Po dokončení kontroly je skript uspán, dokud nedoběhne 30 s dlouhá perioda. Nastavitelným parametrem bude možné určit, zda se má zařízení kontrolovat každých 30 s nebo méně často.

3.2.2 Skript kolektor_snmp

Skript kolektor_snmp periodicky testuje všechny sledované hodnoty (z pohledu SNMP se jedná o jednotlivé OID) u zařízení, která má podle databáze na starosti. Výsledkem je série SNMP příkazů typu get, které jsou postupně poslány a u kterých se čeká po dobu 10 s na odpověď. Možností, jak tento úkol řešit, je několik. Nabízí se postupné vyřízení (odeslání a čekání), pseudoparalelní zpracování vytvořením samostatných procesů, a nebo hromadné odeslání a hromadné čekání na odpovědi. I zde bude možné nastavit délku periody mezi zjišťováním hodnot.

U dat získávaných pomocí SNMP jsem si nadefinoval vlastní 3 typy dat kvůli rozdílnému zpracovaní nebo uložení v databázi. Jedná se o:

1. typ integer – představuje okamžitou hodnotu, například teplotní údaj.
2. typ set – reprezentuje celé číslo, které má přiřazený určitý význam. Tento typ je použit především u stavových hodnot, např. stav zdroje, apod.

3. typ counter – jedná se o rostoucí hodnotu, například průtok dat, apod. Zde není podstatná samotná hodnota, ale rozdíl mezi současnou a předešlou hodnotou naměřenou před zadaným časovým intervalem.

3.3 Návrh databáze

Při návrhu NetSurveyoru bylo nutné věnovat dostatek pozornosti návrhu databáze, která reprezentuje entity a vztahy mezi nimi. Z analýzy jsem vyvodil existenci následujících základních entit v systému:

- Entita zařízení reprezentující reálné zařízení v síti, se sledovanými hodnotami, které nás na něm zajímají.
- Kolektory, které měří nebo kontrolují přidělená zařízení v síti.
- Uživatelé, kteří mají přístup do systému jako uživatel nebo jako administrátor.
- Grafy, které zobrazují určené sledované hodnoty z daných zařízení.
- Statistiky jednotlivých sledovaných hodnot.

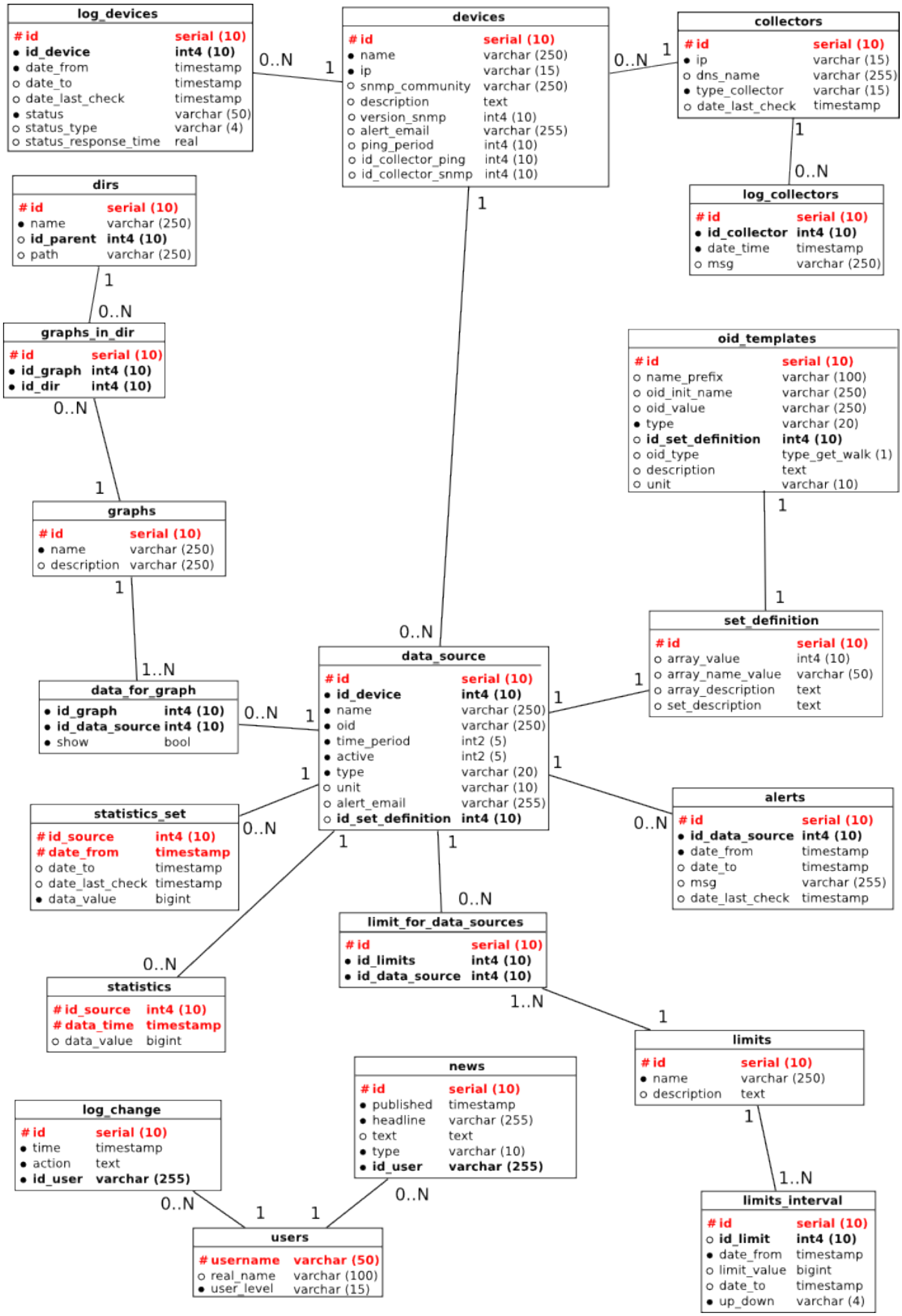
3.3.1 Tabulky

Pro vytvoření reálné databáze bylo dále potřeba promyslet vazby mezi zmíněnými entitami a také promyslet rozklad některých entit podle doporučení k návrhu databáze, tedy využití tzv. 1., 2. a 3. normální formy E. F. Codda. Zohlednit jsem musel také funkční požadavky. Výsledkem tohoto procesu je návrh struktury tabulek na obrázku č. 4.

Tabulka *collectors* představuje jednotlivé kolektory a obsahuje informaci o IP adrese, kde kolektor běží, a typ (snmp nebo ping). Kolektor obecně může sledovat 0 až n zařízení.

Tabulka *log_collectors* slouží k uchování stavových zpráv z kolektorů. Příkladem může být informace o spuštění či vypnutí, nebo případné chybové zprávy. Chybové zprávy týkající se připojení k databázi jsou z pochopitelných důvodů zapisovány do souboru collector.log.

Tabulka *devices* uchovává informace o sledovaném zařízení jako IP, verzi SNMP, heslo ke čtení SNMP dat (community string). Každé zařízení vložené do tabulky je periodicky kontrolováno na dostupnost pomocí ICMP-paketu a informace o délce této periody udává sloupec ping_period jako násobitel minimálního intervalu 30 s. Nechybí možnost definování úrovně případných poplachů a emailové adresy pro zasílání varování. Jméno a popis slouží především informačnímu účelu.



Obrázek 4: Návrh struktury databáze

Účel tabulky *log_devices* spočívá především v uchovávání chybových zpráv, například zpráva o nedostupnosti (time out), zpráva o tom, že se nezdařila autorizace v rámci SNMP, apod. Řádky tabulky, kromě informace, kdy se stav objevil poprvé, také poskytují údaj do kdy stav trval a kdy byl naposledy kolektorem zkontrolován.

Tabulka *data_source* slouží k uchování informací o jednotlivých sledovaných hodnotách. K tomu je potřeba znát OID, interval kontroly (číslo krát 30 s), typ dat OID (integer, set nebo counter). Dále je zde opět určení úrovně poplachů *alert_level* a email pro zasílání poplachů *alert_email*. Pro pohodlnější práci je možné vložit do systému zařízení se všemi OID, tedy sledovanými hodnotami, které lze měřit, ale pokud nás aktuálně nezajímají, můžeme je deaktivovat ze sledování pomocí sloupce *active*.

Tabulka *set_definition* představuje definici množin stavů sledovaných hodnot typu set. Jedná se například o stav napájecího zdroje na zařízení, stav větráků nebo teplotních čidel.

Informace uchovávané v *oid_templates* mají význam především při vkládání nového zařízení do systému. V průvodci přidání zařízení existuje fáze, kdy systém otestuje, zda na zařízení existují OID, která nás obvykle zajímají. Právě tyto zajímavé hodnoty ke sledování jsou řádky z této tabulky. Za pozornost stojí sloupec *oid_type*, který rozlišuje typ get a typ walk. Typ get znamená, že se má testovat jen zadané OID. Typ walk udává, že nás zajímá celý podstrom daného OID. To je vhodné například pro různé ukazatele spojené se síťovými rozhraními na routerech apod.

Tabulky *statistics* a *statistics_set* představují hlavní úložiště naměřených hodnot pomocí SNMP-protokolu. *statistics* je určena pro záznam snmp hodnot dříve představených typů integer a counter. Kromě hodnoty se ukládá čas měření. Do tabulky *statistics_set* se ukládají sledované hodnoty typu set.

Význam tabulky *limits* spočívá v uchování informace horní nebo dolní mezní hodnoty, při jejímž překročení dojde k vyvolání poplachu. Pro bližší představu se jedná například o maximální datový objem konektivity, atd. Problém u těchto kontrolních hodnot je, že se v průběhu času mohou měnit (navýšení smlouvy, apod.). Tyto případy řeší tabulka *limits_interval* představující samotnou hodnotu spouštějící poplach doplněnou o časovou informaci od kdy a do kdy. Pokud není vyplněn sloupec *date_to*, bere se nastavení jako časově neohraničené do budoucnosti. Takto jsou reprezentovány aktuální mezní hodnoty. Zároveň je v tabulce *limits_interval* uvedena informace zda se jedná o dolní nebo horní mez. Informace mezních hodnot se vážou ke zdrojům dat v tabulce *data_source* ve vztahu N:N pomocí spojovací tabulky *limit_for_data_sources*.

Tabulka *alerts* představuje záznamovou tabulku, kde jsou uchovány informace o překročení mezních hodnot poplachů. Záznam v tabulce *alerts* uchovává časové informace od kdy do kdy problém nastal.

Řádky v tabulce *graphs* reprezentují grafy, které se vykreslují ve webovém rozhraní. Graf se skládá z křivek, které vizuálně zobrazují jednotlivé sledované hodnoty identifikované v tabulce *data_source*. Informace, které zdroje dat se mají v daném grafu zobrazit, jsou uloženy v tabulce *data_for_graph*. Samotná data se pak berou z tabulek *statistics* a *statistics_set*.

Pro větší přehlednost je grafy možné organizovat do adresářů, které představuje tabulka *dirs*. Adresáře je možné uspořádávat do stromové struktury pomocí sloupců *id_parent* a *path*. Informace určující který graf patří do kterého adresáře, obsahuje tabulka *graphs_in_dir*. Lze tak kromě adresářů dle zařízení vytvořit adresář s konkrétními grafy, které zajímají uživatele.

Tabulka *users* představuje identifikaci uživatele a jeho úroveň oprávnění. Podrobnější informace o autentizaci a autorizaci naleznete v části věnované implementaci webového rozhraní.

Tabulka *log_change* bude uchovávat informace o změnách v systému. Konkrétněji informace o tom, kdo, co a kdy nastavil nebo změnil.

Tabulka *news* reprezentuje krátké zprávy o novinkách v monitorovacím systému pro všechny uživatele.

3.3.2 Triggery

Momjian [15] píše, že „Triggery (česky též spouště) nabízí alternativní způsob provádění akcí na základě operací INSERT, UPDATE nebo DELETE. Triggery volají serverové funkce pro každý modifikovaný řádek před nebo po provedení operace“.

Pod pojmem trigger se často označuje jak samotné pravidlo za jakých okolností volat uloženou proceduru, tak i samotná procedura. V tomto kontextu termín použiji i já. Triggery lze použít jako validátory před vložením údajů nebo pro modifikaci jiných tabulek po provedení jedné ze jmenovaných operací.

Trigger *sp_log_change* slouží k zaznamenání změn v datech do tabulky *log_change*. Změny se týkají operací vložení, změny nebo mazání a zaznamenány jsou informace čas, kdo a popis operace. Trigger je nastaven ke spuštění po provedení všech tří operací nad následujícími tabulkami: *kolektors, data_source, devices, dirs, graphs, graphs_in_dir, limits, limits_interval, set_definition*.

Trigger *sp_insert_statistics* se provádí po vložení řádku do tabulky *statistics* nebo *statistics_set*. Úkolem tohoto triggeru je kontrola, zda vložená hodnota nepřekračuje nějaký z nastavených poplachů specifikovaných v tabulce *limits* a *limit_interval*. Pokud byl některý z limitů opravdu překročen, vytvoří se záznam (nebo se jen aktualizuje v případě již trvajícího poplachu) do tabulky *alerts*.

3.3.3 Uložené procedury

Uložené procedury jsou volány uživatelem nebo uvnitř jiných procedur. Mohou používat vstupní parametry a vracet data. Lze je použít jako pomocné funkce nebo i složitější části aplikačního kódu. Procedury jsem psal převážně v jazyce PL/PGSQL, který standardní SQL rozšiřuje o proměnné, podmíněné vyhodnocování a cykly.

Procedura *sp_timestamp_to_unix_time* převede datový typ timestamp na vyjádření v unix epoch time.

Procedura *sendmail* je na rozdíl od ostatních psána v jazyce PL/Perl. Funkce umožňuje zasílání emailů a je volána z jiných procedur, především při vyvolání poplachu překročením limitu. Podmínkou správné funkčnosti je, aby operační systém obsahoval správně nakonfigurovanou aplikaci sendmail.

PostgreSQL nenabízí možnost udržovat vnitřní proměnné v rámci jedné relace spojení. Této vlastnosti bych využil především při zaznamenávání změn v nastavení, které provádí trigger *sp_log_change*, kde je potřeba uložit informaci o tom, který uživatel danou operaci provedl. Celý problém jsem obešel vytvořením dvojice procedur *set_active_user* a *get_active_user*. *set_active_user* vytvoří dočasnou tabulku a vloží do ní login uživatele. *get_active_user* je pak volána z triggeru *sp_log_change* pro zjištění loginu aktuálního uživatele.

sp_data_for_curve je procedura starající se o přípravu dat pro vykreslení křivky v grafu. Jejím vstupem jsou id sledované hodnoty, časový interval, který nás zajímá a velikost kroku mezi hodnotami. Procedura vrací tabulku hodnot obsahující čas kroku a hodnotu. *sp_data_for_curve* je příklad agregační funkce pracující i s velkým objemem dat (například při sestavování ročních grafů apod.).

sp_insert_log_devices se stará o vkládání informace o stavu zařízení do tabulky *log_devices*. Informace zde nejsou reprezentovány periodickou sekvencí vkládáním řádků reprezentujících aktuální stav, ale řádky představují interval stavu zařízení. Proto je nutné určité aplikační logiky k vyhodnocení, zda se jedná o stále stejný stav nebo se zařízení dostalo do stavu nového. Právě tento úkol zastává procedura *sp_insert_log_devices*.

Procedura *sp_insert_statistics_set* má na starosti správné vložení dat o sledované hodnotě na zařízení, která je typu set. Význam je podobný jako u procedury *sp_insert_log_devices*, protože i zde se jedná o časové intervaly stavů.

3.3.4 Řešení problému velkého objemu dat v tabulce *statistics*

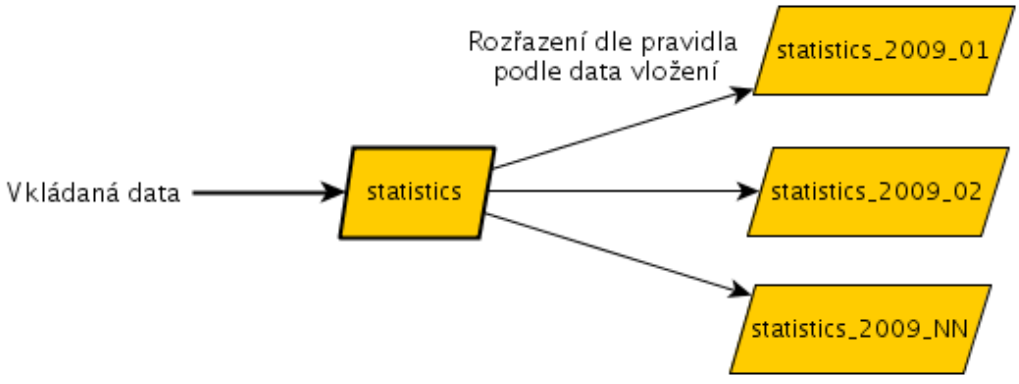
Už během analýzy bylo jasné, že databáze bude uchovávat a pracovat s velkým objemem dat. Především data u sledovaných hodnot typu integer a counter budou rovnoměrně narůstat v průběhu času. Je třeba řešit 2 problémy:

- nárůst objemu dat,
- zpomalení dotazu nad tabulkou *statistics* vlivem velkého objemu dat.

Řešením prvního problému je postupné přepočítávání naměřených hodnot do větších časových intervalů mezi hodnotami v určitém časovém horizontu. Při stáří 2 měsíce budou naměřené hodnoty přepočítány na interval 2 minuty, což odpovídá přesnosti plánovaného denního grafu. Data starší 6 měsíců budou přepočítána na interval 14 minut. Po určité době se nejstarší informace jednoduše zahodí v závislosti na dostupném úložném prostoru. O toto pravidelné přepočítávání se bude starat procedura *sp_deflate* a bude v denní periodě spouštěna aplikací pgAgent.

Zpomalení dotazů nad velkým objemem dat je záležitost, která se musí vyřešit už během návrhu. Naštěstí databázový systém PostgreSQL nabízí elegantní řešení s využitím partitioningu tabulek [16]. Partitioning tabulek využívá pravidel k rozdělení dat při vkládání do několika samostatných tabulek. V SQL dotazech se však stále dotazujeme na původní tabulku.

Hlavním krokem při zavedení partitioningu pro tabulku *statistics* je problém, podle čeho data rozdělit. Nabízí se možnost rozdělit data podle id datového zdroje nebo podle data vložení (například po měsíci). Předpokládám velký počet sledovaných hodnot a velký počet pravidel, která by takto vznikla, mohou podstatně zpomalit zpracování SQL dotazů, jak uvádí dokumentace [16]. Rozumnější volbou bude tedy rozdělit tabulky dle času vložení dat s intervalem po měsíci. Takto můžeme mít aktivní pouze 2 pravidla a to pro tento a následující měsíc.



Obrázek 5: Použití metody partitioning u tabulky statistics

3.4 Webové rozhraní

Základní koncepce vzhledu a členění webového rozhraní je názorně zobrazeno v konceptu rozložení prvků na obrázku 6. V horizontálním směru se aplikace dynamicky přizpůsobuje prohlížeči, ale minimální šířka zobrazení je 780 px. Ve vertikálním směru má horní část s menu pevně danou výšku a oblast aktuálního pohledu se dynamicky mění v závislosti na zobrazovaném obsahu.

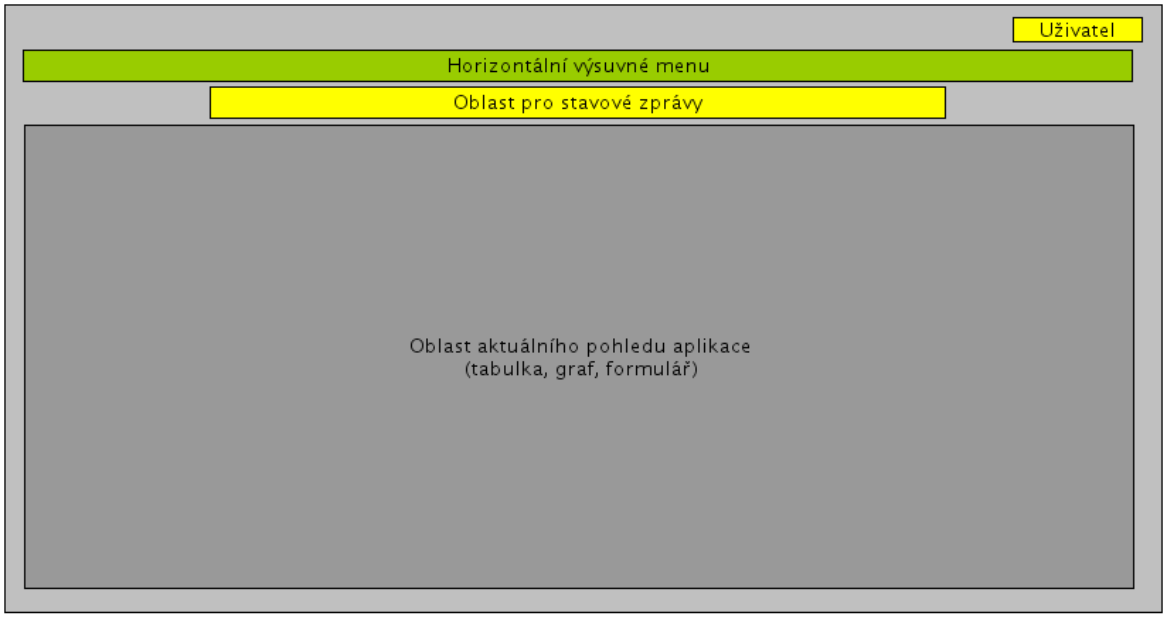
Oblast aktuálního pohledu aplikace je místo, kde se zobrazují informace nebo ovládací prvky v závislosti na právě prováděné akci. Nejčastěji jde o tabulky, grafy a formuláře.

V oblasti pro stavové zprávy se zobrazují informace o úspěchu, či neúspěchu dokončené operace. Například zpráva, zda smazání vybraného zařízení proběhlo v pořádku, případně chybová zpráva, že nejde odebrat kolektor, který má nastaveno nějaké zařízení ke sledování.

Horizontální výsuvné menu slouží k přístupu k funkcím a uživatelům.

Oblast Uživatel informuje o jménu a typu aktuálně přihlášeného uživatele. Dále poskytuje tlačítko k odhlášení (v závislosti na použitém způsobu autentizace).

Nápověda představuje odkaz, který zobrazí nápovědu v novém okně prohlížeče.



Obrázek 6: Rozložení prvků webového rozhraní

3.4.1 Pohledy a funkce pro uživatele

V následující podkapitole vyjmenuji navrhované pohledy a funkce dostupné přes webové rozhraní. Nejprve se budu věnovat pohledům dostupným všem uživatelům.

– **Hlavní stránka** – zobrazuje přehledové a statistické informace, jako je počet sledovaných zařízení, sledovaných hodnot, spuštěných poplachů a funkčních kolektorů. Dále obsahuje výpis novinek, kde se uživatel může dozvědět o novinkách v systému.

– **Monitoring**

  – **Přehled monitoringu** – pohled zpřístupňující stav kolektorů, nedostupné zařízení a tabulku poplachů. Stránka se sama obnovuje v intervalu 60 s.

  – **Grafy (adresáře)** – adresářové zobrazení dostupných grafů. Pohled je vhodný pro rychlý přístup k vybranému grafu.

  – **Grafy (výběr) –** uživatel definuje, které dostupné grafy ho zajímají a v jakém časovém období si je přeje zobrazit. Pohled je vhodný k sestavení přehledu hodnot, které uživatele zajímají.

– **Záznamy**

  – **Zařízení –** tabulkový přehled zařízení v systému s možností zobrazení podrobných informací.

  – **Kolektory –** tabulkový přehled všech kolektorů v systému se zobrazením aktuálního stavu.

– **Nápověda** – zobrazuje uživatelskou nápovědu.

3.4.2 Pohledy a funkce pro administrátora

Administrátor oproti uživateli má možnost využít další 2 podsekce menu.

– **Záznamy**

  – **Změny v systému –** zobrazení změn provedených ostatními administrátory a samotným systémem.

– **Konfigurace**

  – **Zařízení**

    – **Seznam zařízení** – tabulkový přehled všech zařízení v systému s možností editace a mazání.

    – **Vložit nové zařízení (s průvodcem)** – průvodce vložením nového zařízení, které umožňuje poloautomatické nadefinování sledovaných hodnot i vytvoření grafů.

    – **Vložit nové zařízení –** vložení zařízení bez průvodce, vhodné pokud u nového zařízení chceme pouze sledovat dostupnost.

  – **Grafy**

    – **Seznam grafů** – tabulkový přehled všech grafů v systému s možností editace a mazání.

    – **Vytvořit nový graf –** formulář pro vytvoření nového grafu z dostupných sledovaných hodnot.

    – **Adresáře –** pohled umožňující editaci adresářů a reorganizaci grafů v nich.

  – **Sledované hodnoty**

- **Seznam sledovaných hodnot –** tabulkový přehled všech sledovaných hodnot v systému s možností editace a mazání.
- **Vložit nové sledování** – formulář pro přidání nové sledované hodnoty u již dříve vloženého zařízení.

– **Poplachy**

- **Seznam poplachů** – přehled všech nastavených poplachů s možností editace a prohlížení historie.
- **Nastavit nový poplach –** vytvoření nového poplachu nad některou ze sledovaných hodnot.

– **Novinky**

- **Seznam novinek –** přehled již vložených novinek s možností editace.
- **Vložit novinku** – přidání novinky, která se zobrazí všem uživatelům na hlavní stránce.

– **OID šablony**

- **Seznam šablon –** přehled šablon OID používaný v průvodci přidání zařízení. Možnost editace a mazání.
- **Přidat šablonu –** formulář pro přidání šablony.

– **Nástroje**

- **Skener zařízení –** na základě vložení IP adresy a SNMP community string ověří, které OID z tabulky *oid_templates* lze na zařízení zjistit a jakou má aktuální hodnotu.
- **SNMP walk** – funkce vytvoří SNMP příkaz typu walk a zobrazí vrácené informace.
- **SNMP get** – funkce provede zaslání get příkazu protokolu SNMP a zobrazí vrácená data.

– **Nápověda** – rozšířená verze nápovědy pro administrátora.

3.5 Použití knihoven při vývoji webového uživatelského rozhraní

Jeden z argumentů pro výběr PHP jako jazyka pro tvorbu webového uživatelského rozhraní jsem zmínil dostupnost velkého počtu knihoven, které mají zjednodušit samotný vývoj. V této kapitole popíšu vybrané knihovny použité pro realizaci projektu monitorovacího systému NetSurveyor.

3.5.1 PHP framework Nette

David Grudl (autor Nette) [18] tvrdí, že: „Nette Framework je výkonný framework pro pohodlné a rychlé vytváření kvalitních a moderních webových aplikací v PHP 5. Eliminuje bezpečnostní rizika, podporuje AJAX, DRY, KISS, MVC a znovupoužitelnost kódu“.

Nette je mladý původní český framework využívající návrhový vzor MVC (Model-View-Controller). Předností vzoru MVC je oddělení částí kódu dle toho, zda reprezentují data a operace s nimi (model), nebo se starají o zobrazení (view – pohled), případně zda se starají o ovládání a reakce na uživatelské akce (controller – ovladač).

Z mého pohledu vidím výhody v použití Nette v aplikaci MVC (lepší rozdělení a udržitelnost kódu), využití nových objektově orientovaných vlastností jazyka PHP5, několika zajímavých konceptů a aktivní české komunity uživatelů. Za nevýhodu pak mohu označit rychlý vývoj frameworku Nette.

3.5.2 Dibi – knihovna pro práci s databází

Dibi [19] je knihovna pro práci s několika druhy databází napsaná v PHP5 od stejného autora jako framework Nette. Hlavním cílem knihovny je usnadnění a zpřehlednění zápisu SQL příkazů, eliminace výskytu chyb při minimální náročnosti na výkon. Použití podobné knihovny také usnadňuje případný přechod na jiný databázový systém. Argumentem pro volbu Dibi knihovny byla má dobrá zkušenost z dřívějších projektů.

3.5.3 JPGraph – knihovna na kreslení grafů

Způsobů, jak implementovat vykreslování grafů, se nabízelo hned několik. Mohu napsat vlastní PHP řešení využívající GD knihovny [20], využít aplikace Gnuplot [21] nebo použít jednu z dostupných knihoven pod některou ze svobodných licencí (GDChart [22], JPGraph [23], PHPGraphLib [24], pChart [25]). Vlastní implementaci jsem zavrhl, protože úroveň dostupných nástrojů je zcela dostačující.

Při výběru správné knihovny jsem zohlednil licenci, test rychlosti vykreslení grafu ze stejných dat, úroveň dokumentace a možnosti nastavení. Dle zmíněných podmínek nejlépe dopadl JPGraph.

JPGraph je knihovna napsaná objektově orientovaným stylem v PHP4. Pro vykreslení využívá knihovnu GD. Nabízí celou řadu typů grafů, z kterých pro své účely využiji hlavně typ dvourozměrného grafu vykreslujícího spline křivky. Na ose x bude čas, osa y bude představovat získanou hodnotu.

Ve výsledné aplikaci bude možné zobrazovat denní, týdenní, měsíční a roční pohled grafu. Dále pak bude umožněno zadat vlastní pohled na graf podle zadaného intervalu a přesnosti.

3.5.4 JQuery – JavaScriptová knihovna

Webové rozhraní bude tvořeno sérií dynamicky vygenerovaných HTML stránek posílaných ze serveru do klientského prohlížeče pomocí protokolu HTTP. Takže celá logika aplikace běží na serveru a vyžaduje komunikaci dotaz – odpověď. U některých uživatelských ovládacích prvků je

však vhodné docílit okamžité reakce, jinými slovy, část aplikační logiky přenést na stranu klienta, tedy do webového prohlížeče. To umožňuje jazyk JavaScript.

Nicméně, samotný JavaScript trpí několika nedostatky a to například rozdíly v interpretaci v jednotlivých prohlížečích. Rozhodl jsem se předejít případným problémům využitím knihovny jQuery [26]. Hlavním argumentem pro volbu této knihovny byla dřívější kladná osobní zkušenost.

4 Implementace

V následující kapitole podrobněji popíšu některé implementované vlastnosti. Také se zmíním o několika problémech, které jsem během vývoje musel řešit.

4.1 Skripty kolektoru

4.1.1 Implementace skriptu kolektor_ping.pl

První verze skriptu byla implementována voláním externí aplikace ping s vhodnými parametry. To se později při větším počtu sledovaných zařízení pochopitelně ukázalo jako nedostatečné řešení a přistoupil jsem k využití perlovské knihovny POE::Component::Client::Ping [27], která umožňuje paralelní zasílání ICMP-paketů „Žádost o echo“. To se ukázalo jako funkční řešení.

Algoritmus vychází z popsaného postupu v kapitole Návrh. Kromě toho umožňuje paralelní zpracování a pro každý vytvořený „ping“ se při obdržení odpovědi nebo vypršení časového limitu (10 s) volá callback funkce, která provede záznam údajů do databáze.

4.1.2 Implementace skriptu kolektor_snmp.pl

Kolektor pro sběr informací přes SNMP byl implementačně náročnější. Při vývoji jsem použil knihovnu SNMP::Effective. Největší problémy a zdržení mi při vývoji způsobila zmíněná knihovna, která nekontrolovala velikost vytvořeného SNMP paketu a nevracela chybu o špatném sestavení a neodeslání dat. Řešením byla implementace vlastní kontroly.

Skript periodicky kontroluje stav sledovaných hodnot zasíláním SNMP paketů s příkazem get a zpracováním odpovědi. Nicméně příprava i kontrola dat je náročnější než u skriptu kolektor_ping.pl. Při sestavování dat pro SNMP paket se musí připravit OID data, roztřídit dle IP adresy a *time_period* parametru. Dále je zohledněn maximální počet OID na jeden SNMP paket. Takto připravené pakety jsou odeslány na zařízení a skript čeká na odpovědi, které průběžně zpracovává voláním callback funkce. V callback funkci se zohledňuje typ dat. U typu integer nebo set je údaj rovnou zapsán do databáze. U typu counter je vypočítán rozdíl s předešlou zjištěnou hodnotou a do databáze je vložen tento rozdíl.

4.2 Uživatelské webové rozhraní

Jak jsem zmínil v kapitole Analýza, prvotní výběr jazyka Perl pro tvorbu uživatelského webového rozhraní nebyl ideální volbou. Během vývoje jsem postupoval pomalu a ve fázi, kdy byla hotova asi 1/6 práce jsem se rozhodl pro přechod na jazyk PHP5. Následně jsem vybral vhodné knihovny jak popisuji v kapitole Návrh a mohl jsem začít se samotnou implementací. Následující popis v žádném případě není vyčerpávající a pro plné pochopení fungování webové části aplikace je nutné nahlédnout do zdrojových kódů.

Struktura zdrojových kódů souborů uživatelského rozhraní se opírá o koncept použitého frameworku Nette, tedy o návrhový vzor MVC. Pro bližší pochopení fungování frameworku Nette doporučuji prostudovat stránky s dokumentací [28].

Model představují třídy uložené v adresáři *app/models/*. Jedná se o statické třídy zastřešující volání SQL dotazu a práce s daty, například soubor Devices.php obsahuje třídu Devices, která obsahuje mimo jiné statické metody add(), edit() nebo fetchAll() (vrácení všech zařízení v databázi).

Controller se v terminologii Nette nazývá Presenter a zodpovídá za zpracování dotazů od uživatele a sestavení výsledné odpovědi, tedy HTML stránky zaslané zpět uživateli. Presentery se nacházejí v adresáři *app/presenter/*. Pro příklad uvedu třídu GraphsPresenter.php, která se stará o splnění akcí a vytvoření pohledů spojených s grafy, například vytvoření formuláře pro definování nového grafu má na starosti metoda *prepareAdd()*.

Část View je tvořena šablonami, které jsou volány z Presenteru. Šablony jsou uloženy v adresáři *app/templates/* a obsahují logiku starající se o vytvoření finální podoby HTML stránky na základě dodaných dat. Například o vytvoření zmíněného formuláře k definování nového formuláře se stará šablona Graphs.add.phtml. Se šablonami úzce souvisí tzv. helpers, které představují pomocné funkce použité v šablonách. Typickou úlohou helpers je formátování dodaných dat, například překlad řetězce do jiného jazyka (dle zvoleného jazyka) nebo formátování čísla timestamp na čitelné datum.

Dále zde existují komponenty Controls, které obstarávají uzavřenou funkčnost napříč celým modelem MVC. Například o vykreslování tabulek s možnostmi jako řazení dle sloupce, vyhledávání a volby počtu vypsaných řádků se stará komponenta DataGrid.

Samotný framework Nette nabízí celou řadu pomocných funkcí, které například usnadňují tvorbu formulářů. Některé z nich jsem samozřejmě využil i v mém projektu.

Celkově hodnotím pomoc zvolených knihoven při vývoji jako přínosnou. V případě frameworku Nette především oceňuji přehledné členění zdrojového kódu dle účelu, což usnadní další postupný vývoj monitorovacího systému. Na druhou stranu musím říct, že vzhledem k dynamickému a ještě neustálenému vývoji Nette jsem občas musel již napsaný kód přespat tak, aby byl kompatibilní

s poslední verzí knihovny a já mohl využít funkcí, které nově vznikly. Naštěstí ve verzi 0.8 byl stav označen jako stable [29] a tuto verzi používám pro svůj projekt i já.

4.2.1 Implementace autentizace a autorizace

Způsobů implementace autentizace a autorizace je celá řada. Při implementaci jsem musel zohlednit, že systém poběží na serverech společnosti SELF Servis, kde se standardně používá pro autentizaci k službám webového serveru Apache rozšíření mod_auth_mysql. Zmíněný modul obstarává autentizaci na základě ověření hesla s databází uživatelů a příslušností do určité skupiny. Na straně vyvíjeného webového rozhraní pak stačilo implementovat autorizační podmínku, která na začátku každé žádosti o akci nebo pohled vyhodnocuje, zda má přihlášený uživatel dostatečné oprávnění. V případě potřeby však může být přidána vlastní autentizace na úrovni samotné aplikace vytvořením autentizačního handleru, který by implementoval předepsané metody v rozhraní Iauthenticator.

5 Nasazení a provoz systému

V 5. kapitole se věnuji popisu instalace jednotlivých částí systému.

5.1 Instalace

Pro korektní funkčnost celého monitorovacího systému NetSurveyor je potřeba zprovoznit všechny části systému.

5.1.1 Požadavky na provoz databáze

Pro provoz databáze je vyžadován databázový systém PostgreSQL verze 8.3 a novější. Dále vytvořte databázi, pro kterou zapnete podporu jazyků PL/PGSQL a PL/Perlu. Podporu jazyků musíte mít nainstalovanou v operačním systému a zapnutí se provádí příkazem `createlang jmeno_jazyka jmeno_databaze.`

Potřebná struktura tabulek, vč. uložených procedur se vytvoří spuštěním skriptu install_db.sql.

5.1.2 Požadavky a provoz kolektorů

Pro spuštění kolektorů je vyžadován operační systém GNU/Linux s nainstalovaným interpretem jazyka Perl verze 5.10 nebo novější. Dále v systému musí být nainstalovány knihovny:

- SNMP::Effective verze 1.06 nebo novější,
- POE::Component::Client::Ping verze 1.14.

Knihovny je možné nainstalovat z balíkovacího systému zvolené distribuce nebo ručně stáhnout z domovských stránek [17] a [27].

Skripty musí mít nastavené správné údaje pro připojení k databázi. Toto nastavení se nachází v konfigurační části skriptu.

Skript *collector_ping.pl* vyžaduje spuštění pod oprávněním uživatele root. Důvodem je využití knihovny POE::Component::Client::Ping, která potřebuje pracovat se sockety typu RAW.

5.1.3 Požadavky a provoz webového rozhraní

Pro běh aplikační části webového uživatelského rozhraní musíte nainstalovat webový server Apache2 s podporou PHP5. V systému rovněž musí být interpret jazyka PHP5 s knihovnami snmp, pgsql a gd.

Konfigurace webového rozhraní je rozdělena do dvou částí. Nastavení cest k souborům se provádí v souboru *document_root/index.php*. Zde je potřeba definovat absolutní cestu k adresáři *document_root* a pokud dojde ke změně umístění knihoven nebo kódů aplikace, musí se pro správný

chod změnit i tyto relativní cesty vůči adresáři *document_root*. Druhým konfiguračním místem je soubor *app/config.ini*. Zde se nastavují parametry připojení k databázi, mód běhu Nette frameworku (produkční nebo vývojový režim viz dokumentace k Nette [28]).

Posledním krokem je nastavení autentizace. Výchozí instalace počítá s použitím autentizace na úrovni serveru Apache jedním z dostupných autentizačních modulů. Systém pak přebírá informaci o uživateli a přidává vlastní autorizační oprávnění a kontrolu.

6 Závěr

Cílem mé bakalářské práce bylo vytvořit analýzu potřeby univerzálního monitorovacího systému. V další fázi jsem měl za úkol navrhnout vlastní implementaci takového systému, aby pokryl požadavky specifikované v analýze. Implementační část představovala samotný vývoj systému. V závěrečné fázi projektu jsem monitorovací systém NetSurveyor nasadil do testovacího provozu.

6.1 Zhodnocení provozu

Vytvořený monitorovací systém v současné době prochází testováním, které má za úkol ověřit stabilitu a výkonnostní možnosti kolektorů a databáze. Testování je realizováno postupným zvětšováním počtu sledovaných zařízení a hodnot. V současné době systém sleduje 65 zařízení a 1517 hodnot přes SNMP a nebyly zaznamenány žádné problémy nebo nedostatky.

Během testovacího provozu jsem nestihl provést komplexnější testování uživatelského rozhraní s více uživateli. Avšak i pouhá prezentace webového rozhraní spolupracovníkům přinesla hned několik nápadů a postřehů ke zlepšení použitelnosti a vylepšení funkcionality. V další fázi práce na projektu mám v plánu provést 2 uživatelská testování s pracovníky dohledu a to krátkodobé, které by zaznamenalo první dojem a intuitivnost rozhraní a dlouhodobé, které bude mít za cíl najít potřebná vylepšení pro běžné používání v praxi.

6.2 Další vývoj systému

Současnou podobou systém rozhodně nekončí a bude průběžně vylaďován a obohacován o nové funkce. Už během vývoje se objevilo několik nových požadavků, které do implementace už nebyly zahrnuty. Také testovací provoz přinesl několik podnětů, které po vyhodnocení ukázaly nedostatky stávajícího řešení systému NetSurveyor. Zachycené informace budou podrobněji prozkoumány a bude rozhodnuto o jejich prioritě. V zásadě však mohu informace získané zpětnou vazbou rozdělit na krátkodobé plány a dlouhodobé plány.

6.2.1 Krátkodobé plány

V krátkém horizontu je především v plánu postupné navyšování zátěže kolektorů a databáze a případné řešení vzniklých problémů. Z původních požadavků na systém nebyla splněna možnost dynamické práce se sledovanými hodnotami. Tato funkcionalita bude doplněna.

Mezi zřejmější úkoly patří opravení chyb a postupné vylepšování webového rozhraní:

- Použít shodné barvy se starým systémem pro vykreslení velikosti toku dat ve směr ven a dovnitř.

– Lepší vykreslení překrývajících se křivek sledovaných hodnot v grafu, například posunutím jedné z křivek o pixel dolů, čímž bude zachována viditelnost a přesnost grafu to ovlivní jen nepatrně.
– Vylepšení některých přehledových tabulek o odkazy na detailní informace o daném zařízení nebo sledované hodnotě.
– Usnadnění vkládání kontrolních hodnot pro poplachy dle typu sledované hodnoty.

Z plánů na vylepšení monitorovacích funkcí zmíním možnost periodického automatického načítání popisu sledovaných hodnot přímo ze zařízení, jako například popis využití síťového rozhraní z konfigurací routerů Cisco apod.. Cílem je udržet systém aktuální a přehledný.

Dále byl zmíněn požadavek zachytávání SNMP trap zpráv s analýzou a ukládáním do archívu v databázi. Na tuto funkcionalitu by bylo jistě vhodné navázat systém poplachů a zasílání varovných emailů.

Rozšíření monitorovacích schopností se týká i vytvoření kolektorů pro sledování dostupnosti určité služby (webový server, mailový server, databáze, atd.) na daném zařízení, které nelze zjistit pomocí SNMP protokolu. Za úvahu stojí, zda pro tyto účely vytvořit na danou službu a zařízení SNMP server, nebo vymyslet a implementovat vhodné vzdálené testování aplikačním protokolem.

6.2.2 Dlouhodobé plány

Dlouhodobé plány představují náročnější rozšíření systému. Užitečnou vlastností by byla možnost sdružovat zařízení do skupin a v rámci těchto skupin vykreslovat počet aktuálně dostupných zařízení. Podobné řešení by bylo vhodné pro přímé sledování dostupnosti modemů.

Pro zvýšení informační hodnoty i většího uživatelského pohodlí může být vhodné přijít s rozumným způsobem grafické reprezentace topologie sítě.

Další možností dlouhodobějšího rozvoje je využit možností protokolu Netflow a Netflow sond pro monitorování síťového provozu na základě IP toků. To je však úkol už značně přesahující časové možnosti jednoho vývojáře.

6.3 Osobní přínos

Vypracování zadané bakalářské práce mi přineslo především ucelenou zkušenost s kompletním vývojem rozsáhlejšího systému. Pro správné vypracování projektu jsem musel nastudovat problematiku kolem monitorování sítě, především protokol SNMP a jeho využití. Seznámil jsem se databázovými systémy MySQL a PostgreSQL a následně tyto znalosti využil v praktické realizaci. Dále jsem využil celou řadou užitečných knihoven jazyka PHP a Perl.

Z projektu si odnáším i studijní cíl a to prostudovat problematiku vývoje softwaru formou agilních metodik, protože si myslím, že pro vývoj podobného projektu by byly vhodnější než klasický vodopádový vývojový cyklus.

Literatura

[1] *Pojem Internet, Wikipedie* [online] 2009, navštíveno 7. 5. 2009. Dostupné z URL: <http://cs.wikipedia.org/wiki/Internet >

[2] *Stránky projektu Netsaint* [online] 2002. Navštíveno 7. 5. 2009. Dostupné z URL: <http://netsaint.sourceforge.net/>

[3] *Stránky projektu RRD-Tool* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://oss.oetiker.ch/rrdtool/ >

[4] *Stránky projektu NET-SNMP* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://www.net-snmp.org>

[5] *Stránky projektu Cacti* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://www.cacti.net/>

[6] DOSTÁLEK L. – KABELOVÁ A. *Velký průvodce protokoly TCP/IP a systémem DNS.* 4. vyd. Brno: CP Books a. s. 2005, ISBN 80-722-6675-6, str. 135

[7] POSTEL J. *INTERNET CONTROL MESSAGE PROTOCOL,* RFC 792 [online], 1981. Navštíveno 7. 5. 2009. Dostupné z URL: <http://tools.ietf.org/html/rfc792 >

[8] CASE J., FEDOR M., SCHOFFSTALL M., DAVIN J. A *Simple Network Management Protocol (SNMP)*, RFC 1157 [online] 1990. Navštíveno 7. 5. 2009. Dostupné z URL: <http://tools.ietf.org/html/rfc1157 >

[9] CASE J., MCCLOGHRIE K., ROSE M., WALDBUSSER S. *Introduction to version 2 of the Internet-standard Network Management Framework*, RFC 1441 [online] 1993. Navštíveno 7. 5. 2009. Dostupné z URL: <http://tools.ietf.org/html/rfc1441>

[10] HARRINGTON D., PRESUHN R., WIJNEN B. *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*, RFC 3411 [online] 2002. Navštíveno 7. 5. 2009. Dostupné z URL: <http://tools.ietf.org/html/rfc3411>

[11] MCCLOGHRIE K., ROSE M. *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, RFC 1213 [online] 1991. Navštíveno 7. 5. 2009. Dostupné z URL: <http://tools.ietf.org/html/rfc1213>

[12] LEGG S. *Generic String Encoding Rules (GSER) for ASN.1 Types*, RFC 3641 [online] 2003. Navštíveno 7. 5. 2009. Dostupné z URL: <http://tools.ietf.org/html/rfc3641>

[13] GILFILLAN I. *Myslíme v MySQL 4: knihovna programátora*. 1. vyd. Praha: Grada Publishing a.s. 2003 ISBN 80-247-0661-X, str. 231

[14] *PostgreSQL 8.3.7 Documentation, Chapter 37. Procedural Languages* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://www.postgresql.org/docs/8.3/static/xplang.html>

[15] MONJIAN B. *PostgreSQL - praktický průvodce*. 1. vyd. Brno: Computer Press 2003 ISBN 80-7226-954-2, str. 177

[16] *PostgreSQL 8.3.7 Documentation, 5.9. Partitioning* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://www.postgresql.org/docs/8.3/static/ddl-partitioning.html>

[17] THORSEN J. H. *Stránky knihovny SNMP::Effective* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://search.cpan.org/~jhthorsen/SNMP-Effective-1.06/lib/SNMP/Effective.pm >

[18] Grudl D. *Stránky projektu Nette* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://nettephp.com/cs/>

[19] Grudl D. *Stránky projektu Dibi* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://dibiphp.com/>

[20] *Image Processing and GD* [online, 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://cz2.php.net/manual/en/book.image.php>

[21] *Stránky projektu Gnuplot* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://www.gnuplot.info/ >

[22] *Stránky projektu GDChart* [online] 2004. Navštíveno 7. 5. 2009. Dostupné z URL: <http://www.fred.net/brv/chart/ >

[23] *Stránky projektu JPGraph* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://www.aditus.nu/jpgraph/ >
[24] *Stránky projektu PHPGraphLib* [online] 2008. Navštíveno 7. 5. 2009. Dostupné z URL: <http://www.ebrueggeman.com/phpgraphlib/>
[25] *Stránky projektu pChart* [online] 2008. Navštíveno 7. 5. 2009. Dostupné z URL: <http://pchart.sourceforge.net/ >
[26] *Stránky projektu jQuery* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://jquery.com/>
[27] CAPUTO R. *Stránky knihovny POE::Component::Client::Ping* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://search.cpan.org/~rcaputo/POE-Component-Client-Ping-1.14/Ping.pm >
[28] *Dokumentace k projektu Nette* [online] 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://nettephp.com/cs/dokumentace>
[29] GRUDL D. *[rev. 300] Stable a development version, verze 0.9* [online], 2009. Navštíveno 7. 5. 2009. Dostupné z URL: <http://forum.nettephp.com/cs/1688-rev-300-stable-a-development-version-verze-0-9>

Seznam příloh

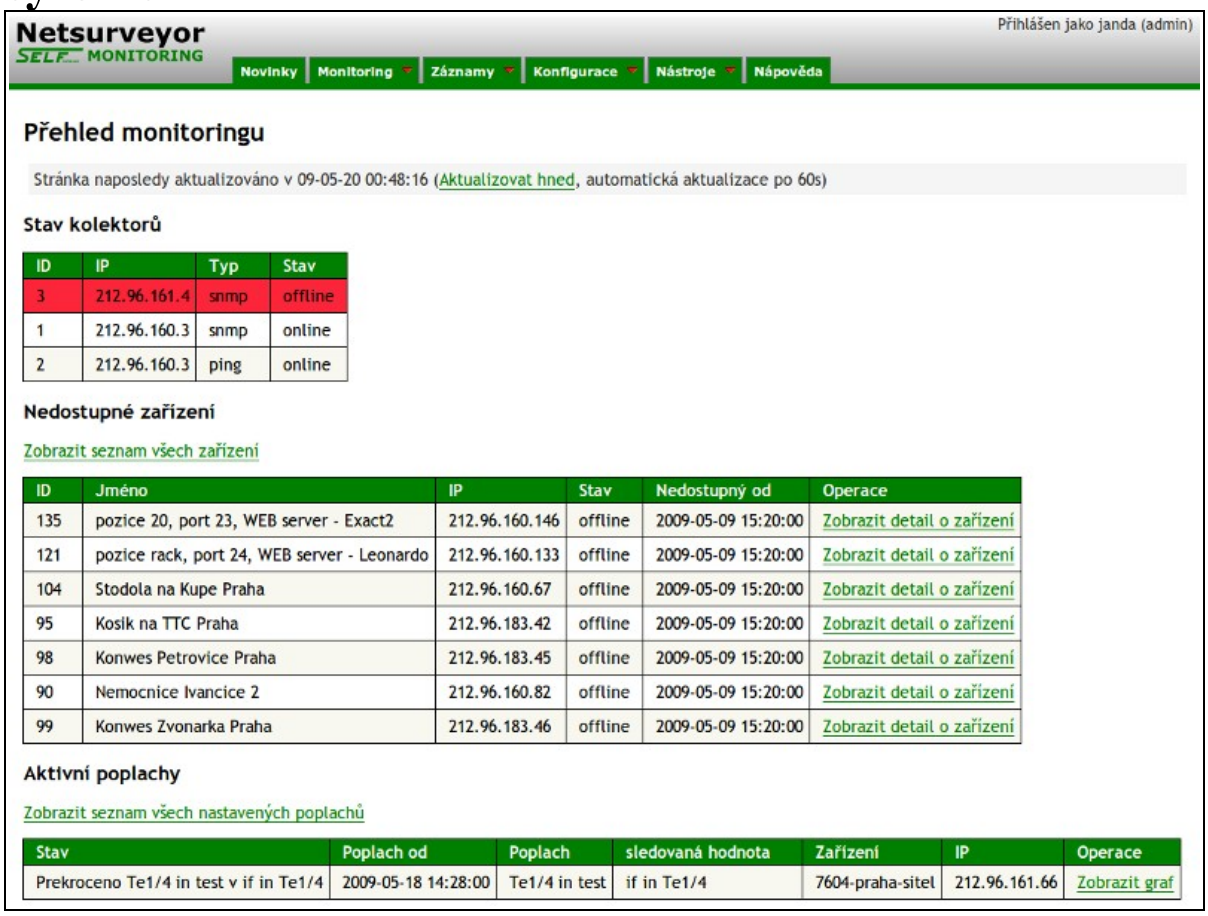
Příloha 1: Přiložené CD
Příloha 2: Ukázky webového uživatelského rozhraní sytému a vykreslovaných grafů.

Příloha 1: Přiložené CD

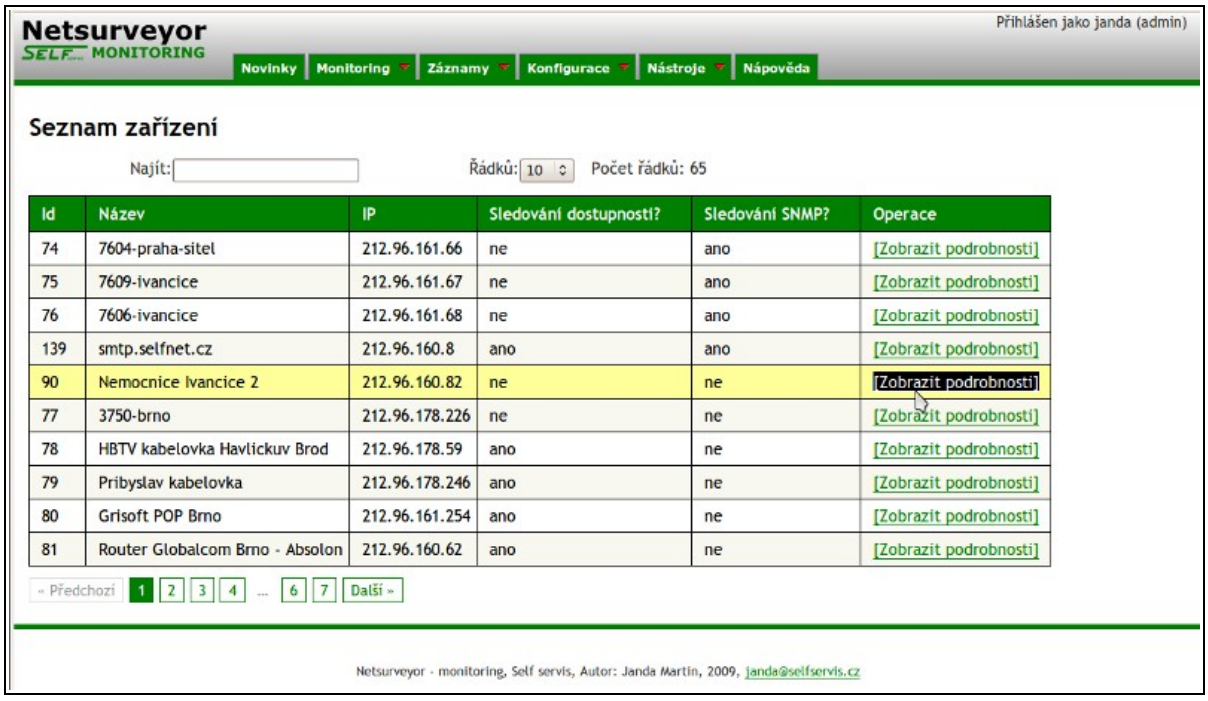
Součástí práce je CD obsahující tyto soubory:

- text bakalářské práce ve formátu PDF a ODT,
- zdrojové soubory kolektorů,
- zdrojové soubory webového rozhraní,
- inicializační SQL script pro vytvoření databáze tabulek, včetně uložených procedur.

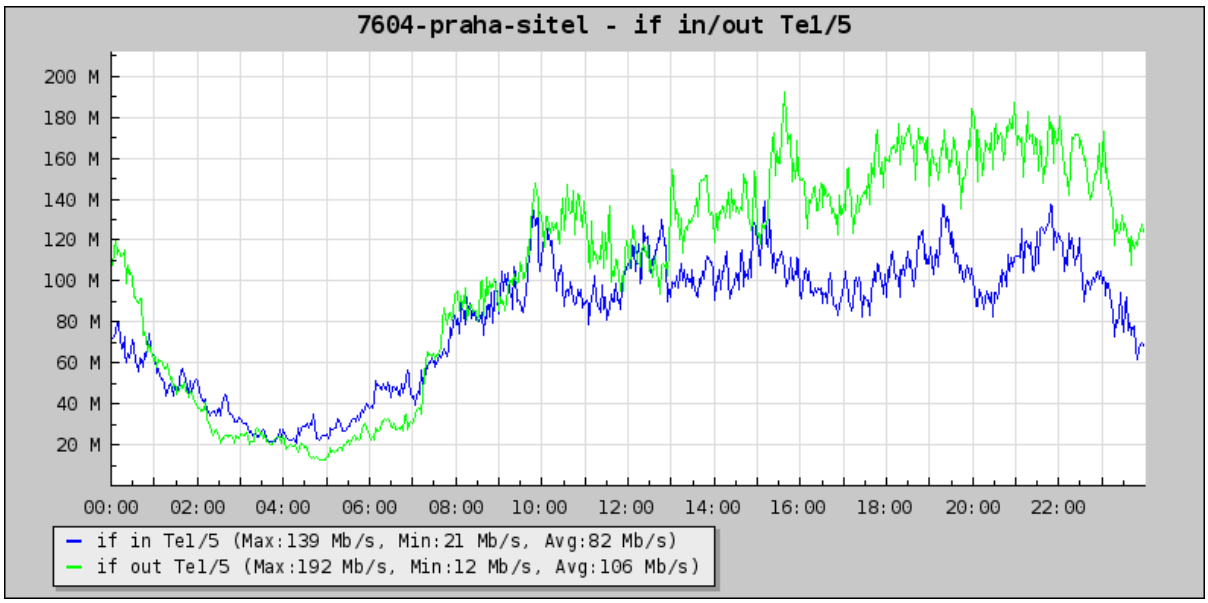
Příloha 2: Ukázky webového uživatelského rozhraní sytému



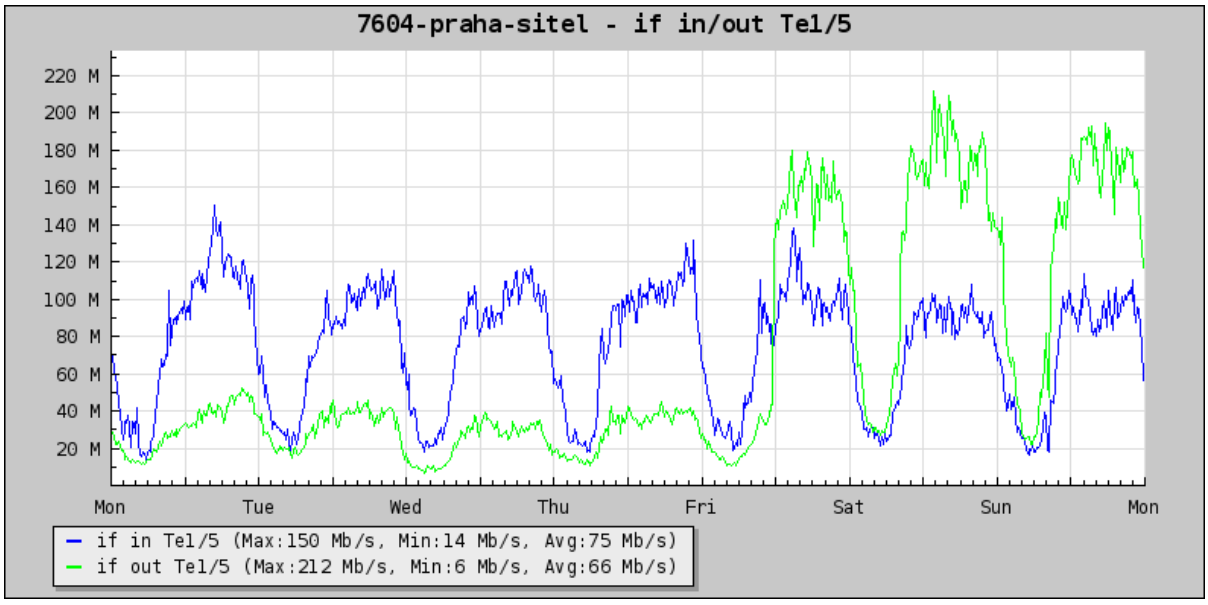
Příloha 2a: stránka Přehled monitoringu



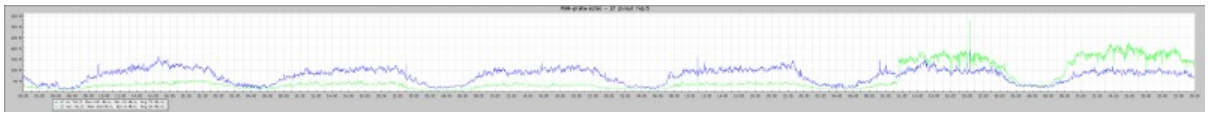
Příloha 2b: stránka "Seznam zařízení"



Příloha 2c: Ukázka denního grafu průtoku dat



Příloha 2d: Ukázka týdenního grafu průtoku dat



Příloha 2e: Ukázka týdenního grafu průtoku dat s větší přesností